FI Anwendungsentwicklung / Duales Studium

Programmierübung (mit Musterlösung)

C/Python–Programmierung / Simulation / Nash–Gleichgewichte

1 Szenario II

In diesem Szenario gibt es 2 Firmen, die ein identisches Produkt auf den Markt bringen wollen. Die Kosten für jede Firma zur Herstellung eines Produktes sind für jede Firma identisch ($c = 3/2$).

Die Preise, zu denen die Firmen ihre Produkte auf den Markt bringen möchten (p_1 und p_2), stehen noch nicht fest. Sie sollen aber so gestaltet werden, dass jede Firma einen optimalen Preis für sich selbst erzielen kann, der ihren Gewinn unter diesen Bedingungen maximiert, und eine stabile Marktsituation entstehen kann (*Nash–Gleichgewicht*).

Eine Marktanalyse hat die Nachfrage nach diesem Produkt ermittelt

$$D_i(p_i, p_j) = 5 - 2p_i + p_j \qquad i = 1, 2, \qquad i \neq j$$

.

D.h. für jede Firma gibt es die gleiche Nachfragefunktion und daher wird erwartet, dass beide Firmen zu einem gleichen Preis kommen, der aber für jede Firma optimal ist.

Auch hier können nur einmal zur Markteinführung die Preise p_1 und p_2 (simultan) festgelegt werden und bleiben dann unverändert.

2 Aufgabe

Ihre Aufgabe ist es, für ihr Unternehmen 1 einen optimalen Preis so zu kalkulieren, dass das Unternehmen 2 ebenfalls optimale Gewinne machen kann und beide Unternehmen dauerhaft auf dem Markt existieren können.

Dabei bestimmt die jeweilige *Nachfragefunktion*, wie häufig ihr Produkt bei Ihrer Firma erworben wird, abhängig von ihrem Preis p_1 und dem Preis Ihres Konkurrenten p_2.

Folgende Grafik (Abb. 1 auf Seite 2) zeigt die Gewinnfunktion ($G_1(p_1, p_2)$) für das Unternehmen 1, in der die Nachfragefunktion ($D_1(p_1, p_2)$) und die Kosten ($c = 3/2$) einberechnet sind:

$$G_1(p_1, p_2) = D_1(p_1, p_2)(p_1 - 3/2) = (5 - 2p_1 + p_2)(p_1 - 3/2) \tag{1}$$

- Ermitteln Sie den maximalen Gewinn, den beide Unternehmen jeweils erzielen können.
- Ermitteln Sie die Anzahl der vorgenommenen Käufe gemäß Marktanalyse zu diesen Preisen.
- Ermitteln Sie den dazu notwendigen optimalen Preis.
- Zeichnen Sie den Punkt in die Grafik ein.

Version vom 2024-12-29

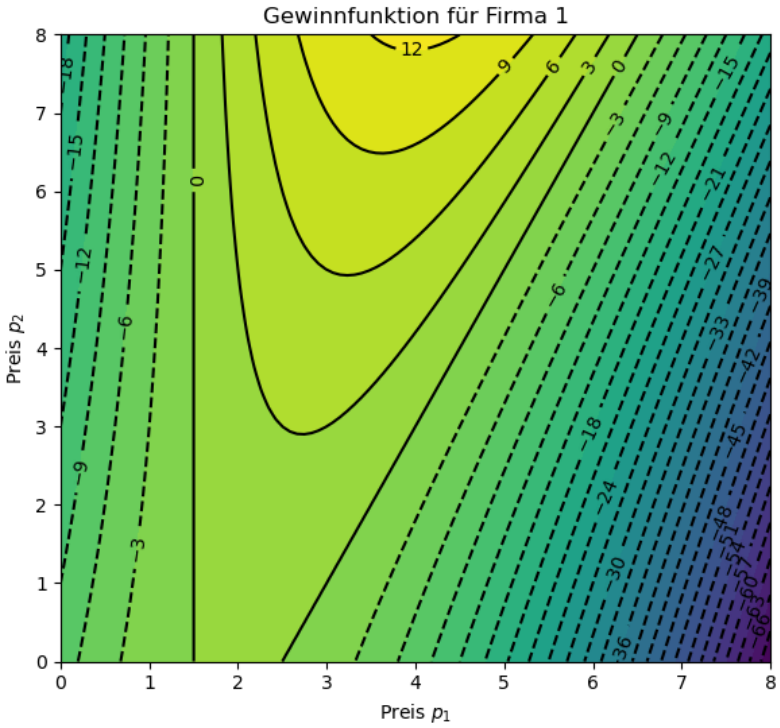

Abbildung 1: Gewinnfunktion aus Sicht des Unternehmens 1

3 Hintergrund

Die Aufgabe beschreibt ein sogenanntes *Nash–Gleichgewicht*, bei dem beide Parteien eine optimale Strategie anwenden, wenn der Gegner ebenfalls bei seiner Strategie bleibt (*kontinuierliche Strategien*). Das *Nash–Gleichgewicht* gehört zum Bereich der *Spieltheorie*. Die Aufgabe ist ähnlich wie bei den diskreten *Nash–Gleichgewichten* mit der Bimatrix, nur dass jetzt kontinuierliche Größen verwendet werden.

Dies ist ein klassischer Aufgabentyp in der VWL.

Entdecker dieses besonderen Gleichgewichts ist John Nash[1], Hauptfigur des Filmes *A Beautiful Mind*. In der Volkswirtschaftslehre gehört diese Aufgabenstellung in den Bereich der Mikroökonomie/Spieltheorie.

4 Abgabe und Zeitrahmen

Die Aufgabe ist *optional* und kann jederzeit zwischendurch bearbeitet werden. Interessant ist, wie genau die geforderten Werte (im Vergleich zur rechnerischen Lösung) bestimmt werden können.

[1] John Forbes Nash, Jr., ⋆13. Juni 1928 in Bluefield, West Virginia; †23. Mai 2015 nahe Monroe Township, New Jersey

5 Musterlösung

Beide Szenarien sind typische Aufgaben aus der Volkswirtschaftlehre im Bereich der Mikroökonomie.

Daher lassen sich beide Aufgaben sowohl analytisch lösen als auch per Simulation.

Eine Musterlösung in *Python* zu beiden Szeanarien findet sich zum Ende dieses Papiers.

6 Die analytische Lösung

6.1 Unterschied zwischen den Szenarien

Beide Aufgaben scheinen auf den ersten Blick recht ähnlich zu sein, aber der Unterschied zwischen den Szenarien ist wichtig.

In beiden Szenarien ist die Grundlage immer die Gewinnfunktion[2], die in diesem Problem von den Preisen, der Nachfragefunktion und den Kosten abhängt ($G(D_i, p_i, c_i)$).

$$D_i(p_i, p_j) \qquad i = 1, 2 \quad i \neq j \qquad \text{Nachfragefunktion (Demand function)}$$

Diese Nachfragefunktion kann von *keinem* der beiden Preise, von *einem* oder von *beiden* Preisen abhängig sein.

Im ersten Szenario ist die Nachfragefunktion für beide Unternehmen nur von dem eigenen Preis abhängig. Bei gleichen Kosten ($c_1 = c_2 = c = 3/2$) muss aus Symmetriegründen dann der Preis für beide Unternehmen derselbe sein ($p_1 = p_2 = p$). Daher muss der gesuchte Preis genau auf der Winkelhalbierenden zwischen der p_1–Achse und der p_2–Achse liegen.

Im zweiten Szenario hat jedes Unternehmen ebenfalls die gleiche Nachfragefunktion (daher muss auch hier der gesuchte Extremwert auf der Winkelhalbierenden liegen), aber die Nachfragefunktion hängt diesmal von beiden Preisen ab.

Das gesuchte Extremum ist daher nicht einfach ein Punkt auf der p_1–Achse, der eine vertikale Linie definiert (wie in Szeanrio I), sondern für jeden Preis p_2 gibt es einen anderen Extrempunkt für p_1. Der Preis p_2 ist hier ein Parameter, der die Lage des Extremwertes beeinflusst. Die Linie der Extrempunkte ist hier eine geneigte Gerade.

In beiden Fällen findet man wegen der Symmetrie den gewünschten Punkt maximalen Gewinns in der 2–dimensionalen Gewinnfunktion als Schnittpunkt zwischen der Linie der Extrempunkte einerseits und der Winkelhalbierenden zwischen p_1–Achse und p_2–Achse andererseits.

In der Grafik der Gewinnfunktionen weiter unten (Abb. 2 auf Seite 6) ist daher in jeder Grafik die Linie der Extrempunkte als auch die Winkelhalbierende eingezeichnet.

Der Schnittpunkt dieser beiden Linien definiert dann den Punkt maximalen Gewinns im Nash–Gleichgewicht.

[2] auch *Zielfunktion*, *Auszahlungsfunktion* oder *Nutzenfunktion* genannt

6.2 Szenario I

Hier lautet die Nachfragefunktion für Unternehmen 1

$$D_1(p_1, p_2) = 5 - p_1 = 5 - p_2 = 5 - p$$

Damit erhalten wir als Gewinnfunktion, wenn wir die Nachfrage mit dem Nettogewinn des Produktes $(p_1 - c_1) = (p - c)$ multiplizieren

$$G(p_1, p_2) = (5 - p)(p - c) = 5p - 5c - p^2 + pc = -p^2 + (5 + c)p - 5c$$

Dies ist eine umgekehrte Parabel, die genau ein Maximum für p besitzt. Setzt man für $c = 3/2$ ein, so erhält man den Extremppunkt p_E (nach Ableitung und Nullsetzung)

$$p_E^* = \frac{13}{4}$$

Dieser ist völlig unabhängig vom Preis p_2 und stellt den Preis dar, bei dem die Gewinnfunktion maximal wird. Die Linie des Extrempunktes in der Gewinnfunktion ist hier, wegen der Unabhängigkeit von p_2, einfach eine vertikale Linie, weil sie für jeden möglichen Preis von p_2 gilt.

Aus Symmetriegründen gilt dieser Preis dann auch für das Unternehmen 2 und daher liegt der endgültige Preis beider Unternehmen auch auf der Winkelhalbierenden.

6.3 Szenario II

Hier ist die Nachfragefunktion von beiden Preisen abhängig:

$$D_1(p_1, p_2) = 5 - 2p_1 + p_2$$

Beim zweiten Szenario ist der maximale Gewinn für das Unternehmen 1 sowohl vom Preis p_1 des Unternehmens 1 als auch vom Preis p_2 des Unternehmens 2 abhängig. Dies macht die Lage deutlich komplizierter. Nun müssen wir einen Gleichgewichtspreis suchen, der für beide Unternehmen optimal ist.

Um hier den Extrempunkt der Gewinnfunktion $G_1(p_1, p_2)$ zu finden müssen wir partiell nach der Variablen p_1 ableiten und dabei p_2 einfach als einen konstanten Parameter betrachten. Der Extrempunkt ist dann wie gewohnt durch Nullsetzen dieser Ableitung zu finden.

$$G_1(p_1, p_2) = (5 - 2p_1 + p_2)(p1 - 3/2) = -2p_1^2 + (5 + 3 + p_2)p_1 - \frac{15}{2} - \frac{3}{2}p_2$$

Leitet man partiell ab

$$\frac{\partial G_1(p_1, p_2)}{\partial p_1} = -4p_1 + (5 + 3 + p_2)$$

und setzt gleich Null erhält man die Extremwerte mit p_2 als weiteren Parameter:

$$-4p_1 + (8 + p_2) = 0 \qquad \Rightarrow \qquad p_{1E} = \frac{8 + p_2}{4}$$

Schreibt man das um und betrachtet p_2 als y–Variable erkennt man übrigens die Geradengleichung, auf der sich die Extrempunkte befinden.

$$p_2 = 4\,p_{1E} - 8$$

Eine Gerade mit 4–facher Steigung und -8 Achsenabschnitt ist in der rechten Grafik (Abb. 2 auf Seite 6) eingezeichnet.

Der Schnittpunkt mit der Winkelhalbierenden zwischen p_1–Achse und p_2–Achse ist dann der gesuchte Punkt für den maximalen Gewinn im *Nash–Gleichgewicht*. Rechnerisch findet man ihn durch Gleichsetzen von p_{1E} und p_2:[3]

$$p^*_{1E} = \frac{8}{3}$$

6.4 Weitergehendes

Für den Interessierten an solchen Fragestellungen und Lösungen aus der Spieltheorie:

Dieses Szenario findet man unter dem Begriff *Cournot*[4]*–Nash* bzw. *Oligopol/Duopol* in der Fachliteratur.

Prinzipiell geht man oft wie folgt vor (Rezept für Duopol):

1. Aufstellen der Nachfrage–Funktion
2. Aufstellen der Kostenfunktion
3. Konstruieren der Zielfunktion/Gewinnfunktion!
4. Durch partielles Ableiten ein parametrisiertes Extremum finden ($f(p_2)$)
5. Bei symmetrischen Problemen ist man damit am Ende:

   Da die *beste Antwort/Reaktionsfunktion* $p_1^* \overset{Nash}{=} R_1(p_2^*) = f(p_2^*) \ldots$

   … kann man $p_1^* = p_2^* = p^*$ setzen und auflösen.
6. Bei Nicht–Symmetrischen Problemen:

   Reaktionsfunktion (*Beste Antwort*) aus der Ableitung konstruieren $R_1(p_2) = f(p_2)$
7. Für Nash–Gleichgewicht gilt dann wegen $R_1(p_2^*) = p_1^*$ und $R_2(p_1^*) = p_2^*$:

   $p_1^* \overset{Nash}{=} R_1(p_2^*) \overset{Nash}{=} R_1(R_2(p_1^*))$

   Auflösen nach p_1^* und einsetzen, um p_2^* zu bekommen.

[3] Extremwerte, die Werte im Nash–Gleichgewicht repräsentieren werden üblicherweise mit einem Sternchen (*) gekennzeichnet
[4] Antoine-Augustin Cournot ⋆28. August 1801 in Gray; †31. März 1877 in Paris

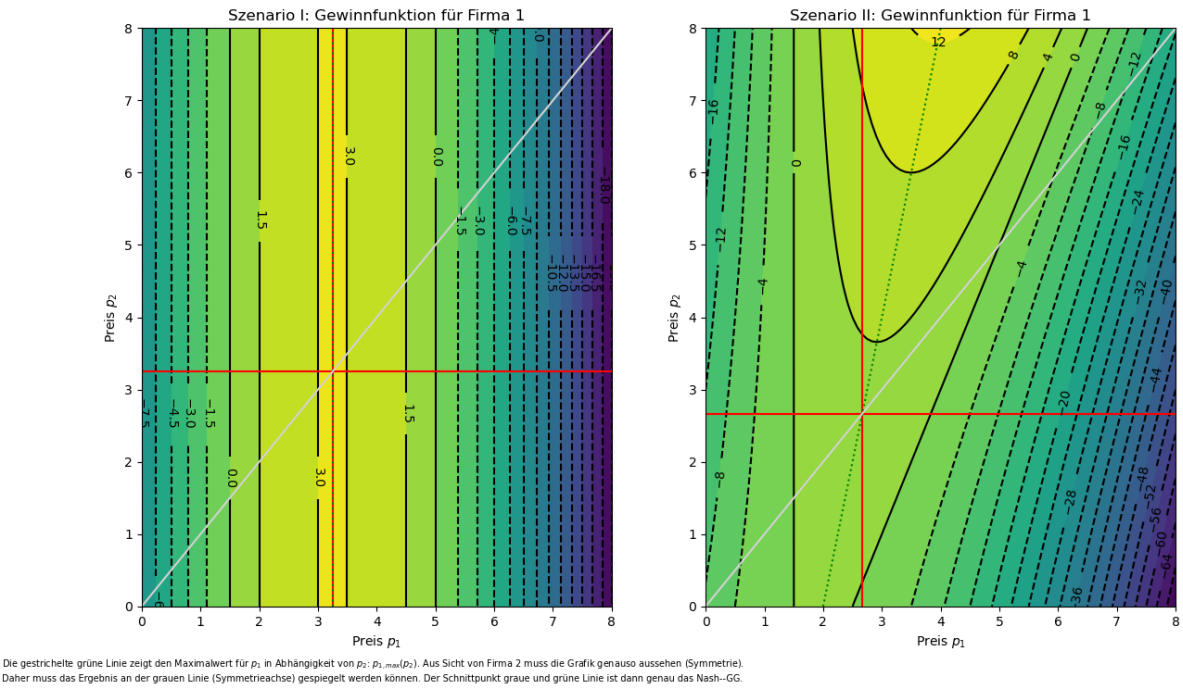

Abbildung 2: Gewinnfunktionen für Szenario I (links) und Szenario II (rechts)

7 Das Programm für Szenario I und II

Musterlösung für `nash.py`

```
# Musterlösung zu den 2 optionalen Aufgaben für FI/Duales Studium
# Es geht um Simulation von Nash-Gleichgewichten
# Author: Frank Zimmermann
# Datum: 2.12.2022
import time
import numpy as np
import matplotlib.pyplot as plt

# Szenario I
# Geschwindigkeitstest
print("Berechnung des Preises und des Maximalgewinnes im Szenario I")
# 1.Version mit Schleife (C)
max1 = 0
p01 = 0
step = 0.001
st = time.time()
for p1 in np.arange(1.5, 5.0, step):
    if (5.0 - p1) * (p1 - 1.5) > max1:
        (p01, max1) = (p1, (5.0 - p1) * (p1 - 1.5))
print("Preis={0:5.4f}, Gesamtgewinn={1:5.4f}".format(p01, max1))
print("Schleife nach C-Art: {0:5.2f} msec".format((time.time() - st) * 1000.))

# 2. Version ohne Schleife (pythonic)
st = time.time()
```

```
xarr = np.arange(1.5, 5.0, step)
nxarr = (5.0 - xarr) * (xarr - 1.5) # Gewinnfunktion
GewinnMax = np.amax(nxarr) # Max finden
PreisMax = (np.where(nxarr == GewinnMax)[0] * step + 1.5)[0] # Preis aus Index
print("Preis={0:5.4f},␣Gesamtgewinn={1:5.4f}".format(PreisMax, GewinnMax))
print("Pythonic␣ohne␣Schleife:␣{0:5.2f}␣msec".format((time.time() - st) * 1000.))

print()

eps = step / 100.0
max2 = 0
p0x2 = 0
p0y2 = 0
for p2y in np.arange(1.5, 5.0, step):
    for p2x in np.arange(1.5, 5.0, step):
        if (5.0 - 2 * p2x + p2y) * (p2x - 1.5) > max2:
            (p0x2, p0y2, max2) = (p2x, p2y, (5.0 - 2 * p2x + p2y) * (p2x - 1.5))
    if abs(p0x2 - p0y2) < eps:
        break
    else:
        max2 = 0

print("Errechnete␣Werte␣für␣Szenario␣I␣/␣Szenario␣II\n")
print("-------------------------------------------------\n")

print(
    "Gesamtgewinn:{0:5.4f}␣/␣{1:5.4f}\nPreis:{2:5.4f}␣/␣{3:5.4f}\n" \
    "Einzelgewinn:{4:5.4f}␣/␣{5:5.4f}\nKäufe:{6:5.4f}␣/␣{7:5.4f}\n".
    format(max1, max2,
           p01, p0x2,
           (p01 - 1.5), (p0x2 - 1.5),
           (5.0 - p01), (5.0 - p0x2)))

print("Analytisch␣berechnet:\n"
    "Gesamtgewinn:{0:5.4f}␣/␣{1:5.4f}\nPreis:{2:5.4f}␣/␣{3:5.4f}\n" \
    "Einzelgewinn:{4:5.4f}␣/␣{5:5.4f}\nKäufe:{6:5.4f}␣/␣{7:5.4f}\n".
      format((13. / 4. - 1.5) * (5. - 13. / 4.), (8.0 / 3.0 - 1.5) * (5.0 - 8.0 / 3.0),
             13. / 4., 8. / 3.,
             (13. / 4. - 1.5), (8. / 3. - 1.5),
             (5. - 13. / 4.), (5. - 8. / 3.)))

levels = 20
plt.subplot(121)
x1, y1 = np.mgrid[0:8:100j, 0:8:100j]
z1 = (5 * x1 - 15 / 2 - 2 * x1 * x1 + 3 * x1 + x1 * x1 - x1 * 3 / 2)
contourset1 = plt.contourf(x1, y1, z1, levels, cmap='viridis')
contourlines1 = plt.contour(x1, y1, z1, levels, colors=('k',))
plt.clabel(contourlines1, inline=1)
plt.hlines(p01, 0, 8, color="red")
plt.vlines(p01, 0, 8, color="red")
plt.plot([13 / 4, 13 / 4], [0, 8], color="green", ls="dotted")
plt.plot([0, 8], [0, 8], color="lightgrey")
plt.title("Szenario␣I:␣Gewinnfunktion␣für␣Firma␣1")
plt.xlabel("Preis␣$p_1$")
plt.ylabel("Preis␣$p_2$")

plt.subplot(122)
x2, y2 = np.mgrid[0:8:100j, 0:8:100j]
z2 = (5 * x2 - 15 / 2 - 2 * x2 * x2 + 3 * x2 + x2 * y2 - y2 * 3 / 2)
contourset2 = plt.contourf(x2, y2, z2, levels, cmap='viridis')
contourlines2 = plt.contour(x2, y2, z2, levels, colors=('k',))
```

```
86 plt.clabel(contourlines2, inline=1)
87 plt.vlines(p0y2, 0, 8, color="red")
88 plt.hlines(p0x2, 0, 8, color="red")
89 plt.plot([2, 4], [0, 8], color="green", ls="dotted")
90 plt.plot([0, 8], [0, 8], color="lightgrey")
91 plt.title("Szenario␣II:␣Gewinnfunktion␣für␣Firma␣1")
92 plt.xlabel("Preis␣$p_1$")
93 plt.ylabel("Preis␣$p_2$")
94 textstr = "Die␣gestrichelte␣grüne␣Linie␣zeigt␣den␣Maximalwert␣für␣$p_1$␣in␣Abhängigkeit␣von␣"\
95           "$p_2$:␣$p_{1,max}(p_2)$.␣" \
96           "Aus␣Sicht␣von␣Firma␣2␣muss␣die␣Grafik␣genauso␣aussehen␣(Symmetrie).\n" \
97           "Daher␣muss␣das␣Ergebnis␣an␣der␣grauen␣Linie␣(Symmetrieachse)␣gespiegelt␣"\
98           "werden␣können.␣" \
99           "Der␣Schnittpunkt␣graue␣und␣grüne␣Linie␣ist␣dann␣genau␣das␣Nash--GG."
100 plt.gcf().text(0.02, +0.01, textstr, fontsize=7)
101 plt.show()
```